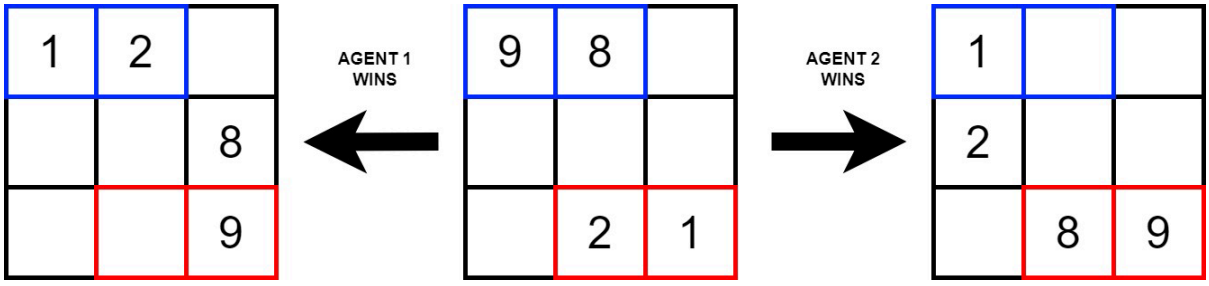
# Assignment 2




In this assignment, you will implement a modified version of the 8-puzzle for 2 agents. Initially the board will have 5 blank tiles and tiles with numbers 1, 2, 8, 9. Agent 1 wins if it can form 1, 2 or 2, 1 on the left of the top row and Agent 2 wins if it can form 8, 9 or 9, 8 on the right of the bottom row (check the figure). The order of the tiles does not matter. The winning positions of agent 1 are represented with blue and the winning positions of agent 2 are represented with red.

**Rules:**

- Agent 1 can only move tiles 1 and 2. Agent 2 can only move tiles 8 and 9.
- Playing the reverse of the previous move is forbidden. If agent 1 moved tile 1 to the right, then agent 1 cannot move it to the left in the next move.
- The game ends when one of the agents wins.
- If an agent keeps its tile in the opponent's same winning tile position for more than 2 rounds, the opponent wins. For example, if agent 1 keeps tile 1 in the bottom right corner for more than 2 rounds, agent 2 wins. By 2 rounds, we mean tile 1 comes to bottom right, agent 2 makes a move, agent 1 moves tile 2, agent 2 makes another move, now agent 1 must move tile 1 or it loses. Note that, in this case tile 1 can be moved to the left to block the other winning position.
- Tiles can only be moved to empty spaces. Each agent can move only one tile at each turn.

Your code will take the initial board configuration from a txt file and write the output to another txt file. Your code will also take the agent number as input (either 1 or 2). This agent will be the selected agent and the other one will be the opponent. Selected agent will start first. Assume that the opponent plays optimally. Maximum tree depth is 10. If the game does not end in 10 moves, it is a draw.

You code is expected to write 3 values to the output file for both parts:

- Minimax value of root node.
  - Win is +1, draw is 0 and loss is -1.
- Number of expanded nodes
- Maximum number of moves that guarantees the win or loss of the selected agent. (**Optional**)
  - If it is guaranteed to win, write the maximum number of moves that guarantees the win.
  - If it is guaranteed to lose, write the maximum number of moves that the loss happens.
  - If a guaranteed win or loss cannot be found, write 0.

Write each one in a different line.

## Part 1

For this part implement a minimax algorithm without α-β pruning. Your code will write the two values to the output file.

We will run your code as follows:

- python3 part1.py <selected-agent-number> <input-filename> <output-filename>

## Part 2

Implement the minimax algorithm with α-β pruning. Again, report the two values.

We will run your code as follows:

- python3 part2.py <selected-agent-number> <input-filename> <output-filename>

**Output Format:**

- Maximum number of moves: Integer
- Number of expanded nodes: Integer

**Important Remarks**:

- The maximum number of moves should be the same for the same initial board configuration for both parts since pruning does not change the result.
- Empty tiles will be represented with 0 in the input file.
- While counting the maximum number of moves, count the moves of both agents.
- For part 1 you are expected to expand the whole tree. For part 2 use depth first exploration. While expanding the nodes, use the following:
  Agent 1 expands tile 1 first, Agent 2 expands tile 8 first. Among the movements of the selected tiles, expand them in this order: Up, Right, Down, Left. For instance, if the board is as the following and it is the agent 2's turn:
  0 8 0
  0 9 1  Expand order (<tile-number>, <move>): (8,R), (8,L), (9,D), (9,L)
  0 0 2

**NOTES:**

- If you have any questions, you can ask using the forum on Moodle. If you think there is anything ambiguous in the description (that can affect the output), please ask. However, before asking, please read the description again.
- In case of an objection, readability of your code is important. If we can't understand your code, we cannot accept your objection. Although not mandatory, documenting your code with inline comments is recommended.

## Cheating Policy

Do not cheat. Do not use ChatGPT generated code.

Any sign of cheating will be penalized, and you will get -50 points for the project, and you will get F grade in case of recurrence in any other project.

## Submission Details

**The zip file you submit should include 2 files: part1.py and part2.py**. There should not be another folder in the zip file, only these files. You will zip them up and submit them on Moodle as a single .zip

file. The name of the zip file should be Cmpe480_Assignment2_<studentid>.zip. **No other type of submission will be accepted.**